DEFAULT NETRANGER ALARM SETTINGS C

This section lists the default NetRanger alarm settings. These settings provide for near-maximum intrusion detection but are not configured to provide automatic response. For each of the ID signatures, string matches, transport events, and policy violations, you can turn on auto-logging or auto-shunning by editing *sensord.conf* or changing the settings through the nrConfigure GUI. The default levels will give you some idea of the usual level of threat for each of these items.

You may want to emphasize certain events at certain sites by increasing the alarm level on the events or by turning on an automatic response. You can also lower the reporting level on certain events to de-emphasize them.

NOTE

The SYN flood detector is configured to work on services that are set up for string match examination. If this is not appropriate for your installation, it may be changed in the file *first.fil* on the BorderGuard. Refer to *Appendix B* for information about editing filter files. Keep in mind that misconfiguring these settings may greatly increase the load on the sensor and its portion of the network bandwidth without providing an increase in protection. Please consult WheelGroup Technical Support for advice on this adjustment.

Table C.1: Intrusion Detection Signatures, Sorted by Default Alarm Level

Alarm Level				
1	**2**	**3**	**4**	**5**
IP options - Bad Option List	IP options - Record packet route	sendmail reconnaissance	Fragmented ICMP traffic	IP options - Loose source route
IP options - Timestamp	ICMP Source Quench	Archaic sendmail attacks	Large ICMP traffic	IP options - Strict source route
IP options - Provide s, c, h, tcc	ICMP Time Exceeded for a Datagram	DNS HINFO Request	smail attack	ICMP network sweep w/Echo
IP options - SATNET id	ICMP Timestamp Request	DNS request for all records	sendmail invalid recipient	ICMP network sweep w/ Timestamp
ICMP Echo Reply	ICMP Information Request (obsolete)	ypserv attempt	sendmail invalid sender	ICMP network sweep w/Address Mask
ICMP Unreachable	ICMP Address Mask Request	ypbind attempt	DNS Zone Transfer from other port	TCP port sweep
ICMP Redirect (change a route)	Unknown IP Protocol	yppasswdd attempt	RPC dump (rpcinfo -p)	Half-open SYN attack
ICMP Echo Request	HTML file has .url link	ypupdated attempt	rexd attempt	WWW phf attack
ICMP Parameter Problem on Datagram	HTML file has .lnk link	ypxfrd attempt	ICMP flood	WWW general cgi-bin attack
ICMP Timestamp Reply	HTML file has .bat link	Sendmail decode alias	WWW .url file requested	TCP Hijacking
ICMP Information Reply (obsolete)		UDP bomb	WWW .lnk file requested	UDP port scan
ICMP Address Mask Reply		mountd attempt	FTP CWD ~root	TFTP passwd file attempt
TCP connection records		FTP SYST command attempted	FTP Improper address specified	Normal SATAN probe

Table C.1: Intrusion Detection Signatures, Sorted by Default Alarm Level

Alarm Level				
1	2	3	4	5
UDP packet records		FTP Authorization Failure	FTP Improper port specified	Heavy SATAN probe
DNS Zone Transfer Request		Telnet Authorization Failure	WWW .bat file request	RPC port registration
		Rlogin Authorization Failure		RPC port unregistration
		POP3 Authorization Failure		Proxied RPC request
				IP options - IP Fragment Attack
				FTP SITE command attempted
				NETBIOS OOB data
				rlogin - froot
				Ident buffer overflow
				Ident newline
				Ident improper request

Table C.2: Intrusion Detection String Matches, Sessions and Alarm Levels

String Match	Session	Alarm Level
"+ +"	Telnet	2
"+ +"	rlogin	2
"RETR passwd"	FTP	4
"/etc/shadow"	Telnet	4
"IFS=/"	Telnet	5
"IFS=/"	rlogin	5
"/etc/shadow"	rlogin	4

Table C.3: TCP/UDP Events and Alarm Levels

Event	Alarm Level
TCP	
Connection request- tcpmux	1
Connection request- echo	1
Connection request- discard	1
Connection request- systat	1
Connection request- daytime	1
Connection request- netstat	1
Connection request- chargen	1
Connection request- ftp-data	1
Connection request- ftp	1
Connection request- Telnet	1
Connection request- smtp	1
Connection request- time	1
Connection request- whois	1
Connection request- dns	1
Connection request- gopher	1
Connection request- finger	1
Connection request- www	1
Connection request- link	1
Connection request- kerberos-v5	1
Connection request- supdup	1
Connection request- hostnames	1
Connection request- iso-tsap	1
Connection request- x400	1
Connection request- x400-snd	1
Connection request- csnet-ns	1
Connection request- pop-2	1
Connection request- pop3	1

Table C.3: TCP/UDP Events and Alarm Levels

Event	Alarm Level
Connection request- sunrpc	1
Connection request- uucppath	1
Connection request- nntp	1
Connection request- ntp	1
Connection request- netbios (port #s 137, 138, and 139)	1
Connection request- imap2	1
Connection request- NeWS	1
Connection request- xdmcp	1
Connection request- nextstep	1
Connection request- bgp	1
Connection request- irc	1
Connection request- imap3	1
Connection request- ulistserv	1
Connection request- exec	3
Connection request- login	3
Connection request- shell	3
Connection request- printer	1
Connection request- courier	1
Connection request- uucp	1
Connection request- pcserver	1
Connection request- kerberos-v4	1
UDP	
UDP traffic - echo	1
UDP traffic - discard	1
UDP traffic - daytime	1
UDP traffic - chargen	1
UDP traffic - time	1
UDP traffic - dns	1
UDP traffic - tftp	3

Table C.3: TCP/UDP Events and Alarm Levels

Event	Alarm Level
UDP traffic - gopher	1
UDP traffic - www	1
UDP traffic - kerberos-v5	1
UDP traffic - sunrpc	1
UDP traffic - ntp	1
UDP traffic - xdmcp	1
UDP traffic - bgp	1
UDP traffic - imap3	1
UDP traffic - ulistserv	1
UDP traffic - biff	1
UDP traffic - who	1
UDP traffic - syslog	1
UDP traffic - printer	1
UDP traffic - talk	1
UDP traffic - ntalk	1
UDP traffic - route	1
UDP traffic - nfs	1

Policy Violations

The sample policy filter *incom1.fil* is designed to identify incoming traffic and pass or fail it based on set conditions. For example, if you specified an FTP server and an SMTP server during the NetRanger configuration, the sample filter would be modified to allow these hosts as exceptions to the rules which fail incoming FTP and SMTP. This sample policy is excellent for use if the NetRanger is protecting the access point between a LAN and an untrusted WAN. If the NetRanger is providing separation between the Finance and Engineering departments at your site, the policy would probably be much looser.

The following policy violations from the sample policy filter *incom1.fil* generate level 3 alarms by default:

```
General tcp failure   tcp elite               udp snmp
General udp failure   tcp ftp data            udp systat
ftp                   tcp imap2               udp tcpmux
telnet                tcp imap3               udp uucp
smtp                  udp netbios             udp uucp rlogin
DNS                   tcp netbios ssn
gopher                tcp netstat
finger                tcp snmptrap
www                   tcp ssh fail
pop2                  tcp
pop3                  tcp at echo
rpc                   tcp at nbp
auth                  tcp at rtmp
nntp                  tcp at zis
ntp                   tcp irc
exec                  tcp kerberos adm
login                 tcp klogin
cmd                   tcp kshell
printer               tcp nfs
ntalk                 tcp systat
uucp                  tcp tcpmux
x11                   tcp uucp rlogin
udp dns               udp
tftp                  udp at echo
udp rpc               udp at nbp
snmp                  udp at rtmp
syslog                udp at zis
ntp                   udp irc
rip                   udp kerberos adm
icmp                  udp klogin
tcp block             udp kshell
dns reply             udp nfs
last_block            udp printer
```

The following policy violation from the sample policy filter *incom1.fil* generates level 4 alarms by default:

```
large icmp fail
```

The following policy violations from the sample policy filters *incom1.fil* generate level 5 alarms by default:

```
IP spoofing
IP source routing
TCP frag header attack
TCP small frag attack
shun
```

DBMS REQUIREMENTS AND SETUP

D

In order for nr.sapd to stage data, you will need access to a database management system (DBMS). Oracle is the default database for the SAP package. Data can also be exported to Remedy's ARS system. If you do not have Oracle or Remedy ARS, you can customize *sapd* to support non-Oracle databases, such as Sybase and Informix.

Oracle Install and Setup

Oracle activation for nr.sapd consists of the following steps:

1. **Plan your Oracle environment.**
2. **Install/Mount the Oracle product.**
3. **Set up environment variables.**
4. **Verify generic connectivity.**
5. **Create user.**
6. **Verify NetRanger–Oracle connectivity.**
7. **Create tables.**
8. **Setup for Remedy ARS (Optional).**

Plan your Oracle Environment

Ask yourself the following questions about your environment:

- Do you have a local or remote database?
- How much disk space is allocated to the database?
- What tablespaces will be used for NetRanger data?

The answers to these questions will allow you to better install/mount Oracle.

Install/Mount the Oracle Product

You have two setup options for the Oracle Product: local or remote. WheelGroup strongly recommends that you export event data to a remote database. Staging data to a database running locally on the Director system could compromise existing monitoring functionality.

Whether you use a local or remote database, you will need to enter passwords for the Oracle users sys and system as part of the install process. You will use these passwords when setting up the Oracle NetRanger user.

NOTE

You will probably have to increase shared memory and semaphore parameters in /etc/system.

Local

Use the Oracle product CD-ROM and install with `orainst`. You will need Oracle Server (RDBMS) and SQL*Plus. Make sure to note the directory location of the install because you will use this in setting `$ORACLE_HOME`. Also note the SID, because you will use this to set `$ORACLE_SID`.

Remote

With the client/server model, Oracle RDBMS runs on a server and the client obtains access to it through proprietary DBMS network facilities, such as Oracle's SQL*NET. If you choose a remote setup for Oracle, you will need to choose one of the following setups:

- Remote homogeneous
- Remote heterogeneous

Remote homogeneous

If your client is the same operating system as the server, you can share the Oracle directory from the server and mount it to the client using normal NFS procedures.

Remote heterogeneous

If your client has a different operating system than the server, use the Oracle Installer with the product CD-ROM to install SQL*Plus and the TCP/IP Protocol adapter. Note the directory location of the install, because you will use this when setting `$ORACLE_HOME`. When the product binaries are available, you will need to configure $ORACLE_HOME/network/admin/tnsnames.ora.

The tnsnames.ora file maps a label (such as remoteDB) to the access protocols required to establish a connection to that database instance. A sample tnsnames.ora follows:

```
remoteDB =
(DESCRIPTION=
(LOCAL=NO)
(ADDRESS=
(PROTOCOL=TCP)
(HOST=DBserver)
(PORT=1521))
(CONNECT_DATA=(SID=nr)))
```

Where `remoteDB =` is the label used in the logon string, i.e., `username/password@remoteDB`. remoteDB is then expanded by SQL*NET into its underlying components:

- HOST=Dbserver
- Port=1521
- Protocol=TCP
- SID=nr

Ensure that the tnsnames.ora file is correctly located and configured; otherwise, you will get an Oracle TNS error message. Oracle will look for it in the following filepaths:

```
/var/opt/oracle/tnsnames.ora
$ORACLE_HOME/network/admin/tnsnames.ora
$HOME/.tnsnames.ora
```

UNIX Setup of Oracle Environment Variables

Oracle applications require a number of different environment variables whose settings depend on the setup you have chosen. Refer to the appropriate section for setting environment variables.

Remote or Local

Whether you are using a remote or local setup, Oracle requires the following three basic environment variables under UNIX: `ORACLE_HOME`, `PATH`, and `LD_LIBRARY_PATH`.

The following environment variables should be added to the .profile for the netrangr user account, and must contain the following information:

- `$ORACLE_HOME` is the base path for the Oracle product, such as /opt/app/oracle/product/7.1.6 or /usr/apps/oracle/app/product/7.3.2. The path usually ends in the version number of your Oracle product.
- `PATH` must contain `$ORACLE_HOME/bin` to allow access to the Oracle executables.
- `LD_LIBRARY_PATH` must contain `$ORACLE_HOME/lib` to allow access to the runtime libraries.

Local Only

If using a local setup, Oracle requires another environment variable, `ORACLE_SID`, which specifies what data area to use with a local database. When this is the case, update `ORACLE_SID` in the .profile as above.

tty problems (HP-UX)

On certain platforms, such as HP-UX, the default tty settings use the character @ for some built-in commands. You will have a problem connecting to a remote database because the tty settings will delete a portion of your connect string username/password@remoteDB.

To examine the tty setting, execute the following command:

```
% stty -a
```

This will show the mappings for commands. If one of the commands, such as kill, is mapped to @, you will need to change it by executing the following command:

```
% stty kill ^Z
```

Verify Generic Connectivity

Verify that Oracle access is enabled.

Before attempting an Oracle connection, ensure that `$ORACLE_HOME`, `$PATH`, and `$LD_LIBRARY_PATH` are set correctly (as described above).

Local

Set `$ORACLE_SID`. The Oracle daemons must be running. Use the Oracle executable `sqlplus` to verify basic connectivity to Oracle with the following command:

```
% sqlplus sys/password
```

If successful, you will enter into interactive SQL. You will see a prompt similar to the following:

```
Connected to:
Oracle7 Server Release 7.3.2.1.0 - Production Release
PL/SQL Release 2.3.2.0.0 - Production
SQL>
```

NOTE

Ideally, you should omit the password string in the command line call. `sqlplus` will then prompt you for the password. This prevents the password from showing up in output from system applications such as `ps`.

Remote

Use the Oracle utility `tnsping` to verify that the basic network setup is correct (this is similar to the standard ping utility) with the following command:

```
% tnsping remoteDB
```

Use `sqlplus` to verify remote connectivity by appending a valid remoteDB string onto the end of the connection string:

```
% sqlplus sys/password@remoteDB
```

Refer to Appendix A: TROUBLESHOOTING for information about the common error messages you might encounter.

Create an Oracle User

Follow these steps before you create a user:

1. **Log into Oracle as sys or system (using `sqlplus` or `sqldba`).**
2. **Determine what Oracle data areas (tablespace) will be assigned for NetRanger usage.**

NOTE

This tablespace needs to be as large as possible. The option of assigning a default tablespace to a user eliminates the problem of having to direct subsequent commands to use a certain tablespace. For high volume NSX sensors, you may need to dedicate a TEMPORARY TABLESPACE and a larger ROLLBACK area to the NetRanger user.

You are now ready to create a user. Follow these steps to create a user:

1. **Create the user with the following command:**

```
SQL>  create user NetRanger identified by PASSWORD default tablespace 'USERS'
```

Where `PASSWORD` is your secret password and `USERS` is your designated tablespace.

2. **Activate the account with the following command:**

```
SQL>  grant dba to NetRanger
```

(Or other lesser privilege based on your site's security policy; at a minimum, the NetRanger user will need privilege to CONNECT, SELECT, INSERT, and DELETE.)

NOTE

Make sure you update the sapd.conf tokens DBUser and DBPass with the new user and password. *sapd* should not be activated until these are set (unless you are customizing the scripts). Use nrConfigure to change these tokens.

Verify NetRanger–Oracle Connectivity

Verify that user netrangr can access the Oracle NetRanger account by running `sqlplus` from the Director system.

Create Tables

Once you have verified Oracle connectivity and the user netrangr, you can proceed with the creation of the NetRanger database tables. These database tables must be created before *sapd* can start loading the data. Begin by connecting to Oracle as user netrangr via `sqlplus` from the /usr/nr/bin/sap/sql/skel directory.

Once connected, execute the following command at the `SQL>` prompt:

```
SQL> @nrdb_master_create
```

NOTE

The first time you use this procedure, you will receive error messages that say that the tables do not exist. This is normal on the first install. Be careful when using these create scripts, because they will drop the existing table, thus deleting all the data.

Setup for Remedy ARS (Optional)

Use schema provided in /usr/nr/bin/sap/skel/nr.alarm.schema.def. Refer to Chapter 5 for more information about Remedy ARS.

Non-Oracle Install and Setup

Please refer to your DBMS software documentation for installation and setup. Use the *Oracle Install and Setup* section as a template.

DBMS Setup Checklist

The following checklist aids in setting up a DBMS. It also provides a high-level overview for Appendix E, in which can be found full details of what follows.

Install and Configure Oracle

_____ Install Oracle Product: CD-ROM or NFS mount

_____ Configure .profile: `ORACLE_HOME, PATH, LD_LIBRARY_PATH`

_____ Set up SQLNET: tnsnames.ora

_____ Verify SQLNET: sqlplus or tnsping

NetRanger–Oracle Setup

_____ Create user

_____ Verify UNIX netrangr user login to Oracle as netranger

_____ Create tables

nr.sapd Setup

_____ Configure /usr/nr/etc/sapd.conf: set DBUser and DBPass

_____ Configure /usr/nr/etc/daemons: add nr.sapd to list

_____ Execute: `nrstart`

_____ Verify that *sapd* is running:

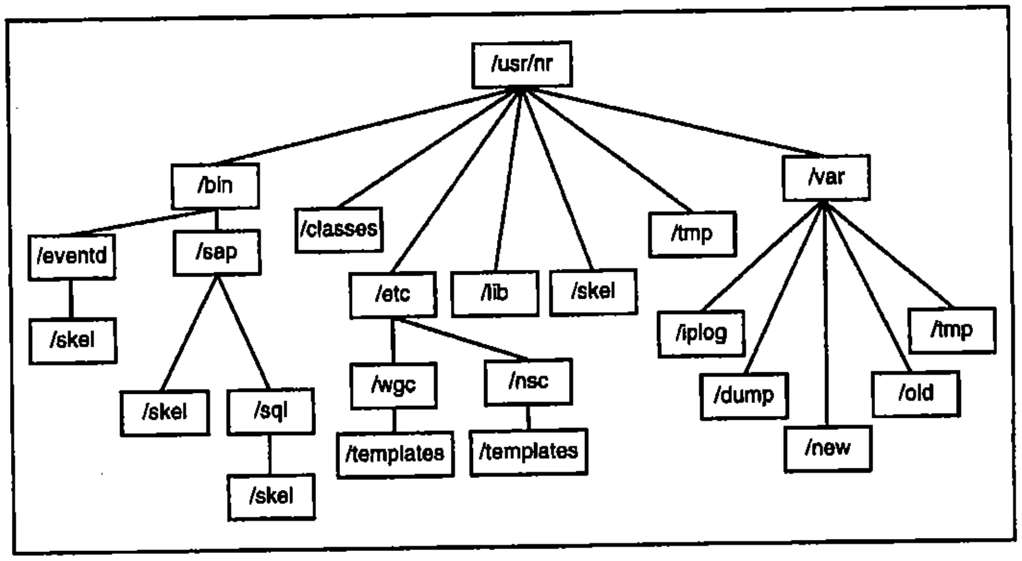
```
tail -f /usr/nr/var/messages.sapd
```

THE NSX FILE STRUCTURE

E

One of the quickest ways to gain a detailed understanding of NetRanger is to review the NSX file structure and underlying elements. A thorough knowledge of the NSX file structure will help you configure and install the NSX as well as help you better utilize NetRanger's features. It should be noted that this file structure serves as the foundation for both the Director and NSX systems.

The default directory location for the NSX subsystem is /usr/nr, which contains the following directory structure:



This appendix provides a brief explanation of each of these directories.

~/bin

This directory contains all of the executables required to run and administer the application services that support an NSX or Director system. These can be loosely grouped into three basic categories:

- Daemon Applications
- Configuration Commands
- System Commands

Daemon Applications

As discussed in Chapter 1, NetRanger is a collection of discrete subsystems that have been implemented via daemons, each with its own well-defined set of services.

NetRanger is also a highly configurable distributed application. The daemon services running on an NSX or Director host depends on the configuration of the application as a whole. By default, an NSX host must have *sensord*, *managed*, *loggerd*, and *postofficed* daemons running. A Director system should have *postofficed*, *smid*, *loggerd*, and *configd* running. In a multi-tiered configuration, one Director system could be configured to only stage data to an RDBMS via *loggerd* and *sapd*, while another Director system could be dedicated to monitoring and response via the *smid* and *configd* daemons.

The full complement of NSX daemons is as follows:

- nr.configd
- nr.eventd
- nr.loggerd
- nr.managed
- nr.postofficed
- nr.sapd
- nr.smid
- nr.sensord
- Nr.InfoServer

Each of these daemon services is briefly described on the following pages.

configd

This daemon reads and writes data to the NSX configuration files (which are currently restricted to **~/etc**). All communication with this daemon is controlled by the following set of SNMP-like executables:

- nrget & nrgetbulk
- nrset & nrunset
- nrexec

NOTE

This daemon provides an *interface* to configuration information. Actual reconfiguration is performed by *managed.* Both command line and graphical user interfaces are provided.

eventd

This daemon is designed to run on Director systems. It allows the Director system to generate user-defined actions for events received by *smid.* A common action is to generate pager notifications via e-mail for alarms of severity 4 and above.

loggerd

This daemon writes out sensor and error data to flat files generated by one or more of the other daemon services. This data is passed to *loggerd* via *postofficed. loggerd* creates two basic types of flat files: a single NSX Event file, and one or more IP Session logs. As noted in the *Overview*, data is written to flat files for reasons of performance and fault tolerance. This data can be staged into an RDBMS and managed for archival via *sapd.*

managed

Whereas *sensord* is responsible for interpreting packet filter events, *managed* is responsible for managing and monitoring these devices. For example, when *sensord* identifies that a certain type of attack should be shunned, it sends a shun command to *managed* via the post office facility. *managed* then reconfigures the packet filter appropriately. Similar commands can be sent to this daemon from the Director. *managed* can also be polled for operational statistics such as:

- The number of network devices being managed.
- The device type.
- Device statistics (packets/sec, bytes/sec, etc.).

postofficed

This daemon serves as the communication vehicle for the entire NetRanger product. All communication between daemons is routed through this service. In most NetRanger configurations the NSX and Director systems reside on separate hosts. Each system relies on a *postofficed* to maintain communication with the other. Note that a *postofficed* is required even when all of the systems are located on a single host.

This daemon service include the following features:

- A proprietary connection-based protocol that resends a message whenever any of its packets are lost.
- Point-to-point routes that may include intermediate post office nodes.
- Communication integrity that is maintained via alternate routes (specified in ~/etc/routes and ~/etc/destinations). When one route fails, communication is switched to an alternate route. Communication is reestablished to the preferred route as soon as it comes back online.

Routing is based on a three-part key that includes the following tuples:

- Organization
- Host
- Application

The Organization IDs are defined in ~/etc/organizations; Host IDs are defined in ~/etc/hosts; Application IDs are defined in ~/etc/services.

sapd

This daemon is a user-configurable scheduler that controls database loading and archival of old event and IP session logs. Oracle or Remedy database loading is accomplished with *sapx*. Data archival is controlled with the native SAP dump scripts.

smid

This daemon routes messages to and from the Director and other daemon services, such as *sensord*. This service relies on routing information stored in the Director's ~/etc/hosts file. In addition to being able to communicate with other daemons, *smid* can redirect messages to other daemon services, such as *eventd* and *loggerd* based on *DupDestination* entries in etc/smid.conf.

sensord

This daemon interprets and responds to all of the events generated by one or more packet filter devices. Data is read in from Network Device sockets and sent to *postofficed* for routing to other daemon services. The types of events received by *sensord* is a function of the filter policy on each packet filter device. The types of messages *sensord* sends to other daemons is based on information stored in ~/etc/sensord.conf. Where these messages are routed is defined in ~/etc/destinations.

Although *sensord* supports 256 levels of alarm events (0-255), only levels 0-6 are currently being used. Level 0 is an internal alarm that *sensord* uses to track such services as RIP, ICMP, and so on; it is never routed to *postofficed*. Levels 1-6 are sent to *postofficed*, and they identify increasing patterns of misuse, where 5 is the highest type of alarm. Note that every alarm message includes an Alarm-ID that identifies a specific attack signature. These alarm signatures are enumerated in ~/etc/sensord.conf as SigOfGeneral entries.

In addition to generating alarm messages for a Director application, *sensord* can automatically instruct *managed* to shun an attack for a specified period of time.

Configuration Commands

The NSX Sensor system comes with the following built-in commands that allow remote devices to configure and gather information about the NSX. As noted in *Operating the Director*, these commands serve as the foundation for the Director's **nrConfigure** interface.

- nrget
- nrgetbulk
- nrset
- nrunset
- nrexec

nrget

This file retrieves single pieces of information from the NSX.

```
nrget socket appid hostid orgid priority token [ identifier... ]
```

nrgetbulk

This file retrieves multiple pieces of information from the NSX.

```
nrgetbulk socket appid hostid orgid priority token [ identifier... ]
```

nrset

This file sets attributes about the NSX.

```
nrset socket appid hostid orgid priority token [identifier...] value [value ...]
```

nrunset

This file unsets attributes about the NSX.

```
nrunset socket appid hostid orgid priority token [identifier...] value [value... ]
```

nrexec

This file executes commands on the NSX.

```
nrexec socket appid hostid orgid priority timeout token [identifier...] value
[value ... ]
```

System Commands

The following scripts are used to perform maintenance on the NSX:

- install
- nrstart
- nrstop
- nrstatus

install

This script is used to install the startup/shutdown files for NetRanger into the UNIX /etc/initd directory. The command takes the following form:

```
            install { add | remove }
```

nrstart

This script starts the NetRanger daemons. It supports the following options:

- -d Don't daemonize
- -h Show help
- -v Show version number

The command takes the following form:

```
            nrstart [-d] [-h] [-v]
```

nrstop

Executing this script stops the NetRanger daemons.

nrstatus

Executing this script returns the currently running daemons.

~/classes

This directory serves as the *rooted class path* for the Director's Java applications, such as **nrConfigure.**

~/etc

This directory contains two types of configuration files: **daemon-specific** and **NSX system** files. All of NetRanger's configuration files are currently ASCII flat files.

Daemon Configuration Files

There is a one-to-one correspondence between daemons and their configuration files. Whereas the naming convention for a daemon is `nr.<daemon>`, the convention for its configuration file is `<daemon>.conf`. Each of these files contains token-based configuration information of the form: `<token> <value>`. For example, the error file for *nr.managed* is identified in *managed.conf* as follows:

```
            FilenameOfError../var/errors.managed
```

Although these entries can be set via a text editor, all changes should be managed via **nrConfigure.** Otherwise, the chance of generating errors increases.

- configd.conf
- eventd.conf
- loggerd.conf
- managed.conf
- postofficed.conf
- sapd.conf
- sensord.conf
- smid.conf

With the exception of *managed.conf*, *sensord.conf*, and *smid.conf* these files currently contain little more than timing intervals and the name of the ~/var/error file for each daemon service. The remainder of this section is devoted to explanations of *managed, sensord*, and *smid.conf.*

managed.conf

This file is used by the NetRanger NSX to configure the management daemon that controls actions on the router. Two types of tokens in this file require special attention: *NetDevice* and *NeverShunAddress*.

NetDevice identifies the **IP address** and the **system password** of an NSG BorderGuard the NSX Sensor is connected to. A separate *NetDevice* entry is required for every BorderGuard system being managed by the NSX system. The format of this token is as follows:

```
NetDevice ROUTER_IP_ADDRESS NSG_Borderguard_v1  ROUTER_PASSWORD
```

NeverShunAddress identifies the IP address of a host or network device you never want the NSX or BorderGuard to shun. At the very minimum you make sure that this file contains *NeverShunAddress* entries for the local system and the remote NSX or Director system. Other addresses can be added based on your operational needs. Example entries might look as follows, where <255.255.255.255> identifies the subnet mask for the IP address you do not want shunned.

```
NeverShunAddress ROUTER_IP_ADDRESS      255.255.255.255

NeverShunAddress NSX_IP_ADDRESS         255.255.255.255

NeverShunAddress INTERNAL_NETWORK       255.255.255.255
```

NOTE

NeverShunAddress entries do not bypass existing filters, but rather prevent them from being completely blocked out of the router.

sensord.conf

This file serves as the heart of the NSX Sensor system, and is the largest of all of the configuration files. It can be logically broken down into the following sections:

- General
- Event Specification
- Strings To Look For
- Matched String and Event Levels
- TCP/UDP Ports and Event Levels
- Policy Violations
- Excluded Events

General

At a minimum, you will need to edit the following lines:

```
RecordOfInternalAddress INTERNAL_NETWORK_ADDRESS INTERNAL_NETWORK_NETMASK

RecordOfInternalAddress  ROUTER_EXTERNAL_IP_ADDR   255.255.255.255
```

These two lines establish which networks NetRanger will protect. **It is important to include all internal addresses because this information is also used for intrusion detection.** You should have one entry for each network address and one for the router external interfaces.

```
RecordOfDataSource      ROUTER_IP_ADDRESS    255.255.255.255
```

This line determines which BorderGuard the sensor daemon will receive information from. You need to fill in the router IP address for this line.

```
RecordOfLogAddress    LOGGED_NETWORK_ADDRESS    255.255.255.0
```

This line establishes a binary log of a particular host or network. This is most commonly used to support an investigation and creates a binary record of all data originating from a particular source address. Initially, this line will probably be commented out.

```
MinutesOfAutoLog       LOG_MINUTES
MinutesOfAutoShun      SHUN_MINUTES
```

These two lines establish the amount of time that NetRanger will shun or log after a particular alarm is received.

Event Specification

The first task-specific segment is Event Specification, which is based upon the SigOfGeneral token. This token identifies an *action* and *alarm levels* for various types of events. *sensord.conf* contains a number of different `SigOf` tokens, all of which share the same basic format, which is defined as follows:

```
SigOfGeneral  SIGID   ACT   D1   D2   D3   D4   # Description of event
```

NOTE

`SIGID` is a numeric identifier that establishes a logical link to internal signature IDs in the *sensord* daemon, and should be treated as **read-only** information. Any changes or additions to these entries will disrupt NSX functionality.

The first column that can be configured is the `ACT` column. It contains a number that specifies how to respond (an *action*) to a particular event. Valid `ACT` settings include:

```
0 -  no action
1 -  shun
2 -  log
3 -  shun and log
```

The remainder of SigOfGeneral maps alarm levels to each of the destinations specified in /usr/nr/etc/destinations. This makes it possible to uniquely define the severity of an event for every destination the NSX is configured to talk to.

The following SigOfGeneral defines what to do when a `TCP Port Sweep` is encountered. It first states that no automatic action should be taken (`ACT=0`). It also defines that level 5 alarms should be sent to its first three destinations, and a level 4 alarm to the last destination.

```
SigOfGeneral  3001   0   5   5   5   4   # TCP port sweep
```

Strings To Look For

This section defines the strings to look for within a given TCP/IP service. The general format for an entry in this section is the following:

```
RecordOfStringName  StringID  TCP/IP Port  Direction  Num Occurrences  "Matched String"
```

`StringID` establishes the unique ID for that particular string. `TCP/IP port` tells NetRanger which port to watch to look for this string. Note that *first.fil* must have a `Copy_To` entry for this port that copies all of its packets to the NetRanger NSX. `Direction` tells the NSX which direction to look for this string:

```
1 - external-to-internal
2 - internal-to-external
3 - both directions
```

The number of occurrences indicates how many times the NSX must see this string before it alarms. `Matched String` is the actual string to be matched. The exact format is controlled by regular expression (regex) syntax.

Matched String and Event Levels

This section identifies action and alarm levels for the string matches defined in the previous section. `SigOfStringMatch` uses the same format as `SigOfGeneral` described above, and is linked to a `RecordOfStringName` by its `StringID`.

```
SigOfStringMatch StringID  ACT  D1  D2  D3  D4  # Description of event
```

TCP/UDP Ports and Event Levels

This section establishes the alarm levels for any packet, successful or not, with a TCP or UDP destination port corresponding to the entry. The action fields and the event level specification are the same as in previous sections.

The following entry illustrates that all Telnet attempts (Port 23) should have no action taken but should generate a level 2 alarm:

```
<port> <action> <dest1> <dest2> <dest3> <dest4>

SigOfTcpPacket     23       0       2        2        2       2
```

The next entry indicates level 3 alarms for all TFTP (port 69) attempts which logs the packets:

```
<port> <action> <dest1> <dest2> <dest3> <dest4>

SigOfUdpPacket     69       1       3        3        3       3
```

Policy Violations

This section identifies the filter names and IDs for all BorderGuard filters that can pass information to the NSX. In the following example, `Income_1_Filter_Fail` has a filter ID of `1000`.

```
            RecordOfFilterName  1000   Income_1_Filter_Fail
```

Each of these policy filters has a matching SigOfFilterName that identifies its action and destinations. The format is identical to the SigOf- tokens described above. In the following example an `Incom_1_Filter_Fail` event will be logged to the second destination defined in /usr/nr/etc/destinations.

```
<filter_id> <action> <dest1> <dest2> <dest3> <dest4>

SigOfFilterName   1000      2       2       2       2       2
```

Excluded Events

The final section of */usr/nr/etc/sensord.conf* is the Excluded Events segment. This is where alarms can be *locked out* for events at specific source addresses. The standard format follows:

```
RecordOfExcludedAddress  SigID  SubSigID    IP Address
```

`SigID` corresponds to the primary signature established in the Event Specification section that establishes all of the primary IDs. `SubSigID` corresponds to those signatures established in the latter sections, such as those for filtering and string matching.

smid.conf

This file is the configuration file for *smid,* which passes information to the Director *nrdirmap* application for display within the OpenView user interface. The only token that needs to be edited in this file is `DupDestination`, which defines duplicate destinations for the events. This token allows a Director system to forward event information onto other services and systems. In the following example, events, commands, and errors of level 1 and above are forwarded onto *loggerd.* This allows the Director to monitor ***and*** log events based on a single NSX data stream. This in turn, helps reduce the amount of network traffic an NSX must transmit to a Director. Instead of having to transmit two identical (but separate) data streams to `director1.wheelgroup`, the NSX is able to send a single stream that is duplicated on the Director platform.

```
DupDestination  director1.wheelgroup  loggerd  1  ERRORS,COMMANDS,EVENTS
```

As the next example shows, this token can also be used to forward events onto another Director system. In this case, however, only events of level 2 and above are forwarded.

```
DupDestination  director2.wheelgroup  smid  2  ERRORS,COMMANDS,EVENTS,IPLOGS
```

NOTE

The host and organization IDs must match entries defined in /usr/nr/etc/hosts.

NSX System Files

The */usr/nr/etc* directory contains all of the NSX's system files. These files are modeled after Unix */etc* files and Unix system administrators should have no problem understanding their format and entity relationships. The files currently are as follows:

- organizations
- hosts
- services
- auths
- destinations
- routes
- daemons
- signatures

NOTE

These files are *described* in order of use rather than alphabetically.

organizations

This file identifies all of the *organizations* and their *organization ids* currently registered for a given NetRanger system. The format of the file is:

```
<org_id><organization_name>
```

A typical file might look as follows:

```
100     companyabc
101     companydef
102     companyghi
1001    companyjkl
```

NOTE

This file serves as a *global* registry that must agree across all of the NSX and Director systems within a NetRanger domain.

hosts

This file is much like a UNIX /etc/hosts file. It enumerates the organizations and hosts that are recognized by a given NSX or Director system in a NetRanger configuration. Each entry has the following format:

```
<host_id>.<organization_id><host_name>
```

where `host_id` and `organization_id` are numeric identifiers assigned by WGC, and *`host_name`* is a DNS-like name assigned by the user. In addition to a list of recognized hosts, this file always contains a *localhost* entry.

A typical hosts file might look like the following:

```
10.100    localhost
10.100    admin.companyabc
11.100    dev.companyabc
12.100    data.companyabc
13.100    nsx-1.companyabc
14.100    nsx-2.companyabc
15.100    nsx-3.companyabc
```

NOTE

Entries for each host must be consistent across *all* NSX and Director systems. Otherwise, some systems will generate "unrecognizable host" messages in ~/var/errors.postofficed.

services

This file defines the *application_id* for each of the daemon services found in **~/bin**. When combined with the *host_id* and *organization_id*, these identifiers uniquely identify a daemon within a NetRanger security map. Each entry has this form:

```
<application id><daemon name>  [<comment>]
```

All NSX hosts should have the same default services file, which is currently defined as follows:

```
10000   postofficed     # Provides the NetRanger comm system
10001   sensord         # Monitors network traffic for security
10002   configd         # Sets and Gets configuration information
10003   managed         # Manages NetRanger components
10004   eventd          # Sends messages and alarms to pagers
10005   loggerd         # Logs events, cmds, errors, IP logs, etc.
10006   smid            # Routes msgs & events to nrdirmap
10007   sapd            # Stages log data into a DBMS
```

auths

This file identifies which type of *configd* operations are authorized for the each of the hosts listed in this file. Each entry follows this form:

```
<host name>        <configd services>
```

In the following example, `admin.wheelgroup` has full authorization, whereas `data.wheelgroup` can only read configuration information.

```
admin.wheelgroup  GET,GETBULK,SET,UNSET,EXEC
nsx-1.wheelgroup  GET,GETBULK
nsx-2.wheelgroup  GET,SET,UNSET,EXEC
```

NOTE

Each entry must have a corresponding entry in ~/etc/hosts.

destinations

NetRanger is a distributed application that has the ability to route messages from a given post office to any of the daemon services registered in the ~/etc/hosts and ~/etc/services files. The destinations file identifies what type of messages get routed to a given application on a given host. Each entry is of the following form:

```
<Destination Id.> <host name> <daemon name> <message level>  <message type>
```

where `<Destination Id>` is a Numeric ID (up to 32 destinations are currently allowed); `<host name>` must be an entry from ~/etc/hosts; `<daemon name>` must be an entry from ~/etc/services; `<message level>` identifies the level of messages that will be routed (messages below that level are not passed on to *postofficed*); `<message type>` identifies the types of messages to route.

For example, in the following destinations file, level 2 (and higher) messages of the type *event, error, command,* and *IP_log* are being routed to the *smid* daemon on the host *admin.wheelgroup*. This type of configuration would normally be found on an NSX system, where *admin.wheelgroup* is a Director system.

```
1   admin.wheelgroup   smid   2 EVENTS,ERRORS,COMMANDS,IPLOGS
```

routes

This file identifies the actual IP routes that *postofficed* uses to send messages between different hosts. Each entry is of the following form:

```
<host name>  <connection #>  <IP address>  <UDP port>  <Type>
```

`<host name>` must be an entry from ~/etc/hosts. `<connection #>` identifies the priority of this entry relative to the other routes enumerated in this file. The lower the number, the higher the route priority. If there is more than one entry of the same priority, *postofficed* accepts the last entry of that priority. `<IP address>` identifies the **end-point** IP address of the **remote system**. `<UDP port>` identifies the UDP port service to route through on each host. These ports are usually in the 30-50000 range. `<Type>` identifies whether the connection is a dial-up vs. dedicated connection. This field is currently not used.

In the following example, `nsx-2.wheelgroup` serves as the primary route to the IP Address 10.1.7.12, and `nsx-3.wheelgroup` is the secondary route to the same host.

```
  nsx-1.wheelgroup      1      10.1.7.9       45000      1
  nsx-2.wheelgroup      1      10.1.7.12      45000      1
  nsx-3.wheelgroup      2      10.1.7.12      45000      1
```

daemons

This file contains the names of daemons that should be started when the system starts up. Each entry is of the following form:

```
              <daemon name>
```

For example, the following sample entry would start *postofficed*, *configd*, *loggerd*, and *smid* on a Director system:

```
nr.postofficed
nr.configd
nr.loggerd
nr.smid
```

NOTE

Whereas the *services* file identifies all of the daemon services a given NSX or Director system is *capable* of running, the *daemons* file identifies the services to launch upon startup.

signatures

This file associates *signature ids* with *signature names* for NetRanger applications, and is used primarily by Director systems. The format of each entry is in the file is:

```
<signature id> "<signature name>"
```

For example, the following sample entry defines a few attack signatures.

```
1000 "Bad Option List"
1001 "Record Packet Rte"
1002 "Timestamp"
1003 "Provide s,c,h,tcc"
1004 "Loose Src Rte"
1005 "SATNET ID"
```

NOTE

This file should NOT be modified.

~/skel

This directory contains basic UNIX configuration files (such as .profile) that are required to configure remote login accounts.

~/tmp

This is the directory where sockets are created by the different daemons. The names of these sockets are dictated by the parent daemon. *postofficed* creates sockets with names such as mailbox.sensord and mailbox.configd. *configd* creates sockets with names such as socket.command.

~/var

This is the default location for all of the log files generated by *loggerd* and error files for all of the applications listed in the ~/etc/daemons file.

Log Files

Two different types of log files are created by *loggerd:* **Event** and **IP Session** logs.

Event File

Although the name of this file can be changed via nrset, the default name of this file is currently hard-coded in *loggerd*. The name of event logs are based on the date and time a new log began (e.g., log.199607291332).

Message Type	Description
0	Default
1	Command
2	Error (Defined)
3	Command Log (Defined)
4	Event (Defined)
5	IP Log
6	Redirect

The schema for defined message types are as follows:

Error records from *managed* (ERROR)

```
1, MsgType (SInt) nrPROTOCOL_ERROR,
2, ULong RecordID,
3,4, GMT timestamp (String year/month/day, hour:min:sec)
5,6 Local timestamp (String year/month/day, hour:min:sec)
6, ULong addr.ApplID,
7, ULong addr.HostID,
8, ULong addr.OrgID,
9, String ErrorData
```

Command Log records from *configd* (COMMANDLOG)

```
1, MsgType (SInt) nrPROTOCOL_COMMANDLOG,
2, ULong RecordID,
3,4, GMT timestamp (String year/month/day, hour:min:sec)
5,6 Local timestamp (String year/month/day, hour:min:sec)
6, ULong addr.ApplID,
7, ULong addr.HostID,
8, ULong addr.OrgID,
9, ULong src.ApplID,
10, ULong src.HostID,
11, ULong src.OrgID,
12, String LogData
```

Event/Alarm records from *sensord* (PROTOCOL_EVENT)

```
1, MsgType (SInt) nrPROTOCOL_EVENT,
2, ULong RecordID,
3,4, GMT timestamp (String year/month/day, hour:min:sec)
5,6 Local timestamp (String year/month/day, hour:min:sec)
6, ULong addr.ApplID,
7, ULong addr.HostID,
8, ULong addr.OrgID,
9, String from,
10, String to,
11, SInt event->level,
12, ULong event->SigID,
13, ULong event->SubSigID,
14, String protocol,
15, String sourceIpAddress,
16, String destIpAddress,
17, SInt event->SrcPort,
18, SInt event->DstPort,
19, String routerIpAddress,
20, String EventData
```

Session Log Files

In addition to detecting attack signatures, *sensord* is able to monitor the UDP traffic associated with a specific type of attack. For example, *sensord* can be configured to monitor all of the packets associated with an IP spoof. *loggerd* creates a separate log file in ~/var/iplog for each of these monitoring sessions. The name of each session log file is based on the IP address of the attacking host (e.g., ip.log.209.15.163.54).

NETRANGER HARDWARE COMPONENTS F

NSX Sensor Hardware Components

- Rack-Mounted PC w/Solaris x86 2.5
- 166 MHz Pentium© processor (10Mbps)
- 110 MHz UltraSPARC© (100Mbps)
  - wide SCSI
- 32 MB RAM
- Three NSX Versions:
  - NSX 1000
  - NSX 2000
  - NSX 5000
- Four configuration options:
  - access through serial port
  - network access
  - modem dial-up
  - video/keyboard

NSX Versions

NSX 1000

- Pentium 166 NSX
- NSG BorderGuard 1000
    - 2 10BaseT/AUI Ethernet
    - 3 WAN Daughtercards/1 Ethernet
    - ISDN
- Up to 512 Kbps

Figure F.1:The NSX 1000

NSX 2000

- Pentium 166 NSX
- NSG BorderGuard 2000
    - 4 Ethernet
    - 4 WAN
    - 2 Ethernet/2 WAN
- 512 Kbps to 10 Mbps Ethernet

Figure F.2:The NSX 2000

SYM_P_0075250

NSX 5000

- UltraSPARC NSX
- NSG ERS/Passport
  - FDDI
  - Ethernet
  - WAN (T1 and T3)
  - (ATM)
- 10 to 100 Mbps Ethernet

Figure F.3:The NSX 5000

NSG BorderGuard Hardware Components

- The BorderGuard is a bridge/router that links IEEE 802.3/Ethernet networks, and other networks via WAN links (e.g., ISDN, leased lines).
- Includes Packet Control Facility (PCF) for firewall routing
- Includes Bridge Control Facility (BCF) for filtered bridging
- Routing Support
  - TCP/IP
  - DECnet Phase IV
  - Novell IPX
  - Banyan VINES
  - AppleTalk II
  - XNS

UNINSTALLING NETRANGER

G

The following procedure describes how to uninstall NetRanger software.

NOTE

This procedure does not describe how to remove HP OpenView from Director systems. **If you want to remove OpenView**, uninstall the NetRanger software first, then refer to the HP OpenView documentation for the OpenView removal procedures.

If you want to remove the NetRanger Director software but continue to use HP OpenView, you should delete your NetRanger Director data from the OpenView databases before uninstalling the Director. Once you uninstall the Director, there will be no way to remove NetRanger data from the databases, other than by completely removing the databases. To delete data from the databases while the Director is still installed, use the Delete Object menu item from the OpenView user interface.

1. **Login as user** `root`**.**

2. **Copy the NetRanger software removal utility to the /tmp directory by typing:**

   ```
   cp /usr/nr/bin/nrUninstall /tmp
   ```

3. **Run the NetRanger software utility by typing:**

   ```
   /tmp/nrUninstall
   ```

4. **Pick the system that you want to remove (usually, this will be option number 1 - All NetRanger Packages).**

5. **A list of packages to be removed will be displayed. Press** y **to continue.**

6. **For each package, status of the removal process will appear on the screen. Read the information and then press return to continue.**

7. **If you have chosen to remove all of the NetRanger packages, then you will be asked if you want to remove the `netrangr` user and group.**

NOTE

Before you choose to remove the netrangr user, make sure that all netrangr users have logged off first.

8. **Press** y **to remove the netrangr user and group or** n **to preserve them.**

NETRANGER *managed* FAQ

This appendix lists the Frequently Asked Questions (FAQ) for *managed*, the process that the NetRanger system uses to manage the NSX, the Director, and various network devices.

Where is *managed*?

nr.managed, located in /usr/nr/bin, is usually run on the NSX to support network devices such as the BorderGuard and Passport.

How do I configure *managed*?

managed uses its default configuration file, managed.conf, located in /usr/nr/etc, for initialization upon startup. Like all NetRanger daemons, this ASCII file can be modified and *managed* will reprocess it upon a HUP signal. Configuration tokens can also be used to adjust the configuration of *managed* on the fly. The default configuration file, the error reporting file, and other configuration items may be changed via tokens.

What is a network device?

The term network device is used because *managed* supports multiple network devices. The current list of supported devices includes the following:

Hardware Device	Operating Systems	Operational Mode
NSG BorderGuard 1000	3.0 & 4.0	Routing & Bridging (v 4.0 only)
NSG BorderGuard 2000	3.0 & 4.0	Routing & Bridging (v 4.0 only)
NSG Passport	Various	Routing & Bridging

What does a network device do?

A network device must sit within the flow of traffic within a network. This is accomplished by the network device acting as either a router or a bridge. A facility must support the selective copying of certain packets (TCP/IP, IPX/SPX, etc.) to the NSX for analysis. The device must also support the capability to be dynamically reconfigured to enforce a security policy. The BorderGuard and Passport provide this through the use of NSG's NetSentry.

How can I determine how many network devices are currently configured?

Using the token, NumberOfNetDevices, you can perform a **get** operation to return the number of network devices currently configured within *managed*. The following example shows how the *managed* queried is currently working with 4 network devices.

```
netrangr% nrget 10003 10 100 1 NumberOfNetDevices
4
```

How many network devices does *managed* support?

There is no predefined limit to the number of devices supported by *managed* other than that imposed by performance constraints. Up to a dozen network devices can be easily handled with little performance impact to the NSX. Hundreds of network devices may be configured, although WheelGroup has not tested this and does not guarantee that this will work.

How do I configure *managed* to work with a network device?

There are two ways of configuring *managed* to work with a network device: through the configuration file and through **GET**, **GETBULK**, **SET**, and **UNSET** commands. The token used to configure a network device is NetDevice. In the configuration file, the format is the following:

```
NetDevice <IP Address> <Type> <Password>
```

These fields specify the network IP address, the type of network device, and the password that NetRanger should use. If a network device supports user accounts, like the Passport, *managed* currently expects that user account to be for user netrangr. BorderGuards do not support user accounts. Although *managed* will be able to support this successfully if password protection is disabled, WheelGroup recommends against this practice. The type field is used for backward compatibility with the configuration files in NetRanger versions 1.0 and 1.1. *managed* no longer uses this information because it is capable of determining the type of network device by itself. However, the type field MUST BE present or the password field will not be properly accessed. Therefore, the string **`foo`** would be adequate for the type description. Here are some examples:

```
NetDevice  10.1.2.1  NSG_BorderGuard  3j8c7Hke
NetDevice  10.1.6.90 NSG_BorderGuard  pdk6Y5nr
NetDevice  10.1.7.2  NSG_Passport     wG7ki8En
```

Network devices may be added/removed on the fly through **`SET`** and **`UNSET`** operations. The following example uses the commands **`nrset`** and **`nrunset`** to dynamically add and remove devices on an NSX with a hostid of 10 and orgid of 100. The priority of the requests is 1. The NetRanger Director's NSX configuration tool also supports these operations.

```
netrangr% nrset 10003 10 100 1 NetDevice 10.1.4.1 NSG_BorderGuard 3j8c7Hke
Success

netrangr% nrunset 10003 10 100 1 NetDevice 10.1.2.1
Success
```

How can I tell which network devices are currently configured?

Using the NetDevice token, **`get`** and **`getbulk`** operations may be performed to identify which devices are in operation. Here are a few examples using the same NSX as discussed above:

```
netrangr% nrgetbulk 10003 10 100 1 NetDevice
10.1.2.1  NSG_BorderGuard_2000  3j8c7Hke
10.1.6.90 NSG_BorderGuard_2000  pdk6Y5nr
10.1.7.2  NSG_Passport       wG7ki8En

netrangr% nrget 10003 10 100 1 NetDevice 10.1.2.1
NSG_BorderGuard_2000 3j8c7Hke
```

How can I determine what the hardware type is for a network device?

Use the NetDeviceType token to perform **get** and **getbulk** operations to identify what type of hardware a network device is. Here are a few examples:

```
netrangr% nrgetbulk 10003 10 100 1 NetDeviceType
10.1.6.90 NSG_BorderGuard_2000
10.1.2.1 NSG_BorderGuard_2000
10.1.6.2 NSG_Passport

netrangr% nrget 10003 10 100 1 NetDeviceType 10.1.2.1
NSG_BorderGuard_2000
```

How can I determine the operating system version of a network device?

Use the NetDeviceVersion token to perform **get** and **getbulk** operations to identify the operating system version for a network device. Here are a few examples:

```
netrangr% nrgetbulk 10003 10 100 1 NetDeviceVersion
10.1.6.90 4.0
10.1.2.1 4.0
10.1.6.2 BC01S1D

netrangr% nrget 10003 10 100 1 NetDeviceVersion 10.1.2.1
4.0
```

How can I tell if *managed* is properly working with a network device?

Use the NetDeviceStatus token to perform **get** and **getbulk** operations to identify the current status of the communication link between *managed* and the network device. Here are a few examples:

```
netrangr% nrgetbulk 10003 10 100 1 NetDeviceStatus
10.1.6.90 Active
10.1.2.1 Connecting
10.1.6.2 Configuring

netrangr% nrget 10003 10 100 1 NetDeviceStatus 10.1.2.1
Active
```

The valid values of the connection status are as follows:

Status	Description
Inactive	The link is completely down*
Connecting	*managed* is trying to connect via a Telnet connection
Login_Sent	*managed* has sent the username netrangr to the login prompt
Password_Sent	*managed* has sent the password to the password prompt
Configuring	*managed* is configuring the network device
Active	The link is active

*An inactive link means that the password is wrong, it is an unsupported network device, a user account for netrangr does not exist (if one is needed), or some other type of problem has occurred. An inactive link means that *managed* will remove ALL configuration information pertaining to this network device during normal garbage collection operations.

What happens if the password I configured is wrong?

If *managed* determines that the password specified is incorrect, it will generate an error message and log it to both its error file, *errors.managed* and to destinations configured in /usr/nr/etc/destinations. After the error message is generated, the connection state is set to Inactive and *managed* will delete ALL configuration information pertaining to this network device during normal garbage collection operations.

Sample error file message:

```
12/21/96 17:41:44, Wrong username/password for net device 10.1.2.1,
errno = 0
```

Sample error message reported in log:

```
2,1101604,1996/12/21,23:41:44,1996/12/21,17:41:44,10003,10,100,
Wrong username/password for net device 10.1.2.1
```

What happens if the network device specified is not supported?

If *managed* determines that a network device is unsupported, it will generate an error message and log it to both its error file, *errors.managed* and to destinations configured in /usr/nr/etc/destinations. After the error message is generated, the connection state is set to inactive and *managed* will delete ALL configuration information pertaining to this network device during normal garbage collection operations.

Sample error file message:

```
12/21/96 17:41:44, Unsupported net device 10.1.6.244, errno = 0
```

Sample error message reported in log:

```
2,1101601,1996/12/21,23:41:44,1996/12/21,17:41:44,10003,10,100,
Unsupported net device 10.1.6.244
```

What happens if the network device reboots?

If the network device reboots, *managed* will reestablish the Telnet connection once the device is again operational. When *managed* loses connectivity to the network device, it will generate an error message and log it to both its error file and configured destinations.

Sample error file message:

```
12/21/96 17:41:44, Connection lost to net device 10.1.6.244, errno = 0
```

Sample error message reported in log:

```
2,1101601,1996/12/21,23:41:44,1996/12/21,17:41:44,10003,10,100,
Connection lost to net device 10.1.6.244
```

Different timeout values are used for different network devices. *managed* notices that the connection is down if it has not received information from a BorderGuard within 45 seconds or from a Passport within 5 minutes. This is generally enough time for these devices to reboot and configure themselves.

Can I have *managed* tell a network device to execute a command?

You can use the NetDeviceCommand token to perform **EXEC** operations to pass commands directly to a network device over the Telnet connection that *managed* has established. The token requires the IP address of the particular network device to be accessed. Any additional arguments are passed directly on to the network device. The token only works if the Telnet connection is active. An example is shown below:

```
netrangr% nrexec 10003 10 100 1 10 NetDeviceCommand 10.1.2.1 net d app
border System 5> net d app
Apply Points:
point   filter name          protocol
first   FIRST_FILTER         IP
Filters active = 1
COMMAND_FINISHED
```

In this case, the command `net d app` is executed on a network device with an address of 10.1.2.1. The response that is passed back includes a copy of the command line, the results of the command, and a token specifying that the command is finished. *managed* will strip out any blank lines that might be returned by the command. This token can be used with a shell script to provide easy command line access to a network device. The example below, called bg1, will take any arguments to the shell script and pass them on to the specified network device.

```
#!/bin/sh
/usr/nr/bin/nrexec 10003 10 100 1 10 NetDeviceCommand 10.1.2.1 $*
```

Does *managed* have any built-in commands for network devices?

managed provides the basic functionality for reconfiguring a network device to shun (i.e., stop) all traffic flowing from a particular network address. This capability is used to stop traffic from known malicious hosts. The *sensord* process on the NSX uses this capability to provide real-time shunning of network connections. An operator may also use the same function to shun an address from the Director.

How does shunning work on a BorderGuard or Passport?

Shunning will ONLY work on BorderGuards and Passports if a file called *shun.fil* containing a filter called *shun* has been provisioned. The *shun* filter should be applied at the first filter point AFTER filter *netranger* has been called. This same configuration must be used whether the network device is operating as a bridge or a router. It will even work if the device is operating as BOTH a bridge and a router and the filter is applied on the IP first filter point and the bridge first filter point.

Here is an example of a filter first_filter that is applied on the first filter point:

```
filter first_filter
   filter netranger;
   filter shun;
end
```

managed will replace the shun filter by replacing shun.fil. However, an initial filter MUST be provisioned that contains a do-nothing filter called *shun*. Here is an example in the recommended format:

```
filter shun
   any continue;
end
```

What command do I send to *managed* to shun an address?

You can use the ShunHost token to perform an **exec** operation to add an address to the list of shunned addresses. *managed* will configure ALL network devices with active connections to shun this address. A value must also be provided to specify the number of minutes for the address to be shunned. Inserting a value of 1000000 (~ 2 years) effectively shuns an address permanently.

After the number of minutes has expired, *managed* will reconfigure the network devices to unshun the address. If multiple shun commands are sent for a single address, *managed* will keep updating the time value for when the address can be unshunned.

Here is an example of a successful shun command:

```
netrangr% nrexec 10003 10 100 1 10 ShunHost 10.1.1.5 1024
Success
```

managed will reject any address that contains a value of 0 or 255 as one of the four address segments. Examples of addresses that would be rejected include 10.1.1.0, 10.1.0.1, 10.1.1.255, 0.0.0.0, and 10.255.255.255.

How do I obtain a list of shunned addresses?

You can use the ShunHostList token to perform a **get** operation to return a complete list of all currently shunned addresses and the number of minutes left for each one to be shunned. If there are no addresses in the list, an error message will be returned stating that the value was unset.

```
netrangr% nrget 10003 10 100 1 ShunHostList
10.1.1.5 1024
```

How do I unshun an address?

You can use the UnshunHost token to perform an **exec** operation to remove an address from the list of shunned addresses. If the address is within the list, ALL network devices will be reconfigured to unshun the address.

```
netrangr% nrexec 10003 10 100 1 10 UnshunHost 10.1.1.5
Success
```

Can I shun a range of addresses?

Yes, the tokens ShunNet, UnshunNet, and ShunNetList provide the same functionality as the tokens previously mentioned for hosts except that they address entire networks. These tokens are designed to provide network security analysts the ability to completely shun an offending network.

The primary difference for these tokens is that a netmask must also be specified along with the IP address. Here are some examples:

```
netrangr% nrexec 10003 10 100 1 10 ShunNet 10.1.1.0 255.255.255.0 1024
Success

netrangr% nrget 10003 10 100 1 ShunNetList
10.1.1.0 255.255.255.0 1024

netrangr% nrexec 10003 10 100 1 10 UnshunNet 10.1.1.0
Success
```

Note that there are two tables of shunning information maintained by *managed*—one for individual addresses and the other for network addresses.

How do I configure *managed* to NEVER shun a particular address?

You can use the NeverShunAddress token to perform **getbulk**, **set**, and **unset** operations to manage a list of IP addresses and netmasks that *managed* compares with addresses to be shunned. If the shunned address falls within the range of addresses specified by the list, *managed* will NOT shun that address on the network devices.

Addresses may be individual hosts or entire networks as specified with the standard IP netmask. For example, 10.1.100.1 255.255.255.255 would define a single host and 10.1.1.0 255.255.255.0 would define an entire class C network.

```
netrangr% nrset 10003 10 100 1 NeverShunAddress 10.1.100.0 255.255.255.0
Success
```

```
netrangr% nrgetbulk 10003 10 100 1 NeverShunAddress
10.1.100.0 255.255.255.0
10.1.1.201 255.255.255.255
```

WheelGroup recommends that the configuration file for *managed* contains NeverShunAddress entries for the NSX, the network device, and the Director.

What is the hierarchy of shunned addresses, networks, and never shun addresses?

When *managed* receives a request to shun an IP address, it first checks to determine if the address falls within the ranges of addresses specified to never shun. It then checks to see if the address falls within the ranges of network addresses that are currently being shunned. *managed* will only apply the shun if the address is not within either set. If the address has already been shunned, *managed* will simply update its internal timer to remove the shun against that address. Shunning network addresses follows a similar pattern.

The difference between an address not being shunned because it is not supposed to be or an address that is already shunned is in the response. If an address falls within the range of addresses that are never shunned, an error message is returned. If an address falls within the range of network addresses that are already shunned, a success message is returned.

When either list of shunned addresses or networks changes, *managed* will modify the *shun* filter on the network devices to reflect the change.

Can I unshun all shunned addresses and networks at once?

Use the UnshunAll token to perform an **exec** operation to unshun ALL currently shunned addresses and networks. If NeverShunAddress entries have not been inserted for the NSX, Director, and network device—and one of them has been shunned—this command may fail.

How fast does shunning occur?

Shunning a host or network usually occurs within 3 seconds or less. It may take longer if a large number of addresses (> 100) are being shunned. This is due to the speed of the storage device on the network device. The new shun filter needs to be written to the filesystem, and then read in and compiled. The list below specifies the types of filesystems used:

```
BorderGuard 1000 - Flash memory
BorderGuard 2000 - Floppy disk
Passport - Hard drive
```

Is there a limit to the number of addresses/networks that can be shunned?

Use the ShunMaxEntries token to configure the maximum number of addresses that may be shunned. The default value is 100, but this may be increased if required. Caution is advised when increasing this limit. The limit exists for several reasons:

- The file systems of network devices do not have unlimited storage space.
- Every address shunned has a slight performance impact on the routing/bridging speed of the network device. A few addresses have a negligible impact; two hundred would have a measurable impact.
- Denial-of-service attacks may be implemented that cause NetRanger to shun large numbers of addresses. All network security solutions are vulnerable to these types of attacks. These attacks may be compared to an adversary launching hundreds of diversionary attacks to draw attention from the real threat.

The following examples illustrate how to increase the limit:

```
netrangr% nrset 10003 10 100 1 ShunMaxEntries 150
Success

netrangr% nrget 10003 10 100 1 ShunMaxEntries
150
```

What happens to the shunned configuration if the network device reboots?

When the network device reboots, *managed* will reapply the shun information when the device is operational. Systems like the BorderGuard will reestablish shunning because it will compile the *shun.fil* file modified by *managed* when it shunned an address.

Can I disable/enable shunning?

Use the ShunEnable and ShunDisable tokens to perform **exec** operations to enable or disable shunning. When *managed* is first started, shunning is always enabled by default. Shunning can be disabled by using the ShunDisable token. The following examples illustrate how shunning may be disabled and then re-enabled:

```
netrangr% nrexec 10003 10 100 1 10 ShunDisable
Success

netrangr% nrexec 10003 10 100 1 10 ShunEnable
Success
```

All shunning information will continue to be collected by *managed* during the period that shunning is disabled. Once shunning is re-enabled, then *managed* will update the *shun* filters on the network devices.

When would I want to disable shunning?

Shunning is accomplished through a Telnet session that *managed* establishes to the network device. Shunning expects that the connection be at the same prompt as at initial login. This is of primary concern for the Passport. Provisioning mode may be entered by sending the command **nrexec ... start prov**. The Telnet session will then be in provisioning mode and shunning will fail. If *managed* is used to provision a Passport, it is HIGHLY recommended that shunning be disabled during that period.

How do I use *managed* to manage the time on the system?

The following list of tokens may be used to get/set the time. In order to change the current time, *managed* MUST be running with root privileges. *managed* expresses time as the number of elapsed seconds since 00:00 GMT, January 1, 1970.

Token	Operation	Description
CurrentTime	**get**	Gets the current time
SetCurrentTime	**exec**	Sets the current time
AdjCurrentTime	**exec**	Adjusts the current time

Changing the time with a delta of greater than +/- 60 seconds may cause problems in running NetRanger processes. In this case, it is recommended to stop/start the affected processes for proper operation. Smaller changes should be accomplished through the use of the AdjCurrentTime token discussed below.

AdjCurrentTime adjusts the system's notion of the current time, as returned by CurrentTime, advancing or retarding it by the amount of time specified in the `struct timeval` pointed to by delta.

The adjustment is affected by speeding up (if that amount of time is positive) or slowing down (if that amount of time is negative) the system's clock by some small percentage, generally a fraction of one percent. Thus, the time is always a monotonically increasing function. A time correction from an earlier call to AdjCurrentTime may not be finished when it is called again.

These commands may be used in NetRanger Directors that synchronize the clocks of computers in a local area network. Such time servers would slow down the clocks of some machines and speed up the clocks of others to bring them to the average network time. This could be accomplished through the use of cron jobs and shell scripts to synchronize remote NSX systems.

How should the netrangr user account be configured on a Passport?

The netrangr user account should be configured with the following characteristics:

Userid:	NETRANGER
Scope:	network
impact:	debug
nmifs:	local, Telnet, fmip, FTP
Directory:	/

Can *managed* write out other files besides its own configuration file?

The tokens WriteFileHosts, WriteFileServices, WriteFileAuths, and WriteFileDestinations can be used to write out the configuration files in the /usr/nr/etc directory. All NetRanger daemons cache the contents of these files in memory upon startup and refresh the information when they reread their configuration file (via a HUP signal or **exec** token ReadFileConfig). ONLY *managed* has the ability to write out these files. The files would be written out if the information cached in memory was modified using the applicable configuration tokens.

What is the list of *managed* specific tokens?

Token	GET	GETBULK	SET	UNSET	EXEC
NetDevice	X	X	X	X	
NetDeviceStatus	X	X			
NetDeviceVersion	X	X			
NetDeviceType	X	X			
NetDeviceCommand					X
NumberOfNetDevices	X				
ShunMaxEntries	X		X		
ShunEnable					X
ShunDisable					X
ShunHostList	X				
ShunHost					X
ShunNetList	X				
ShunNet					X
UnshunHost					X
UnshunNet					X
UnshunAll					X
NeverShunAddress		X	X	X	
CurrentTime	X				
SetCurrentTime					X
AdjCurrentTime					X
WriteFileHosts					X
WriteFileServices					X
WriteFileAuths					X
WriteFileDestinations					X

Index

A

adding Director entities 4-10
alarms 1-9
alternate loading procedures 5-42
alternate routes 1-11
atomic signature 1-8
attack signatures 1-5, 1-8
auths E-16

B

basic Director functions 4-3
BorderGuard
    BorderGuard installation 3-5
    configuration 3-5
    hardware components F-3
    installation options
        bridge 2-3
        Internet router 2-5
        introduction 2-3
        unable to replace existing Internet router 2-6
            if no unused Class C addresses exist 2-8
            unable to subnet to existing Class C address 2-9
            use a Class B address 2-6
            use an unused Class C address 2-6
    manual reconfiguration for shunning 1-9
    physical and operational specifications 2-13—2-15
    standard files to configure B-1
BorderGuard 1000 1-7
BorderGuard 2000 1-7
BorderGuard installation 3-5—3-9
bridge mode B-7
bridging
    and BorderGuard installation 2-3
    and Data Privacy Facility B-13

C

cable requirements (BG1000/2000 to NSX) 3-7
centralized command and control 1-14
changing IP addresses, hostnames, and NetRanger IDs 4-28
changing the event error notification user 4-60
checking the status of the Director processes 4-5
commands
    exec 1-18
    get 1-17
    getbulk 1-17
    set 1-18
    unset 1-18
communication
    alternate routes 1-11
    based on three-part address system 1-11
composite signature 1-8
configd E-3
configuration commands
    nrexec E-7
    nrget E-6
    nrgetbulk E-6
    nrset E-6
    nrunset E-6
configuration of netrangr user account H-14
configuring an application with nrConfigure 4-50
configuring new users 3-25
configuring the user interface
    HP-UX and Sun Solaris 3-21
    IBM AIX 3-24
content-based signatures 1-6
context-based signatures 1-6
copy_to/log_to auditing 1-7
creating a trouble ticket 4-28

D

daemon configuration files E-8
daemons
    applications E-2
        configd E-3
        eventd E-3
        loggerd E-3
        managed E-4

postofficed E-4
sapd E-5
sensord E-5
smid E-5
as fundamental design principle 1-4
daemon configuration files E-8
managed.conf E-9
sensord.conf E-10
smid.conf E-13
daemons at startup E-17
determining the version of a remote daemon 4-26
eventd 1-20, 4-55—4-60
advanced setup
changing the event error notification user 4-60
monitoring multiple organizations 4-59
basic setup 4-56
event script setup 4-56
eventd.conf setup 4-58
overview 4-55
smid.conf setup 4-58
postofficed 1-11
remote configuration 4-23
Data Privacy Facility 1-3
and bridging B-13
filters B-10
pubkeys.dpf B-10
sleeves.cmd B-12
sleeves.dpf B-11
DBMS setup checklist D-8
defining security policy 3-27
deleting Director entities 4-19
destinations E-16
Director
adding entities 4-10
manually adding a Director symbol 4-13
manually adding an application symbol 4-10, 4-12
manually adding an NSX collection 4-14
manually adding an NSX machine symbol 4-10
basic Director functions 4-3—4-5
starting the Director 4-3
stopping the Director 4-4
centralized command and control 1-14
changing IP addresses, hostnames, and NetRanger IDs 4-28
changing registration files 4-29—4-34
changing symbol, object, and submap characteristics 4-36—4-47
changing the number of events displayed in event list 4-35

checking the status of the Director processes 4-5
command line parameter examples 4-35
configuring new users 3-25
configuring the user interface
    HP-UX and Sun Solaris 3-21
    IBM AIX 3-24
creating a trouble ticket 4-28
deleting entities 4-19
hardware and software requirements 3-10
installation 3-17
introduction 1-3, 4-1
limiting access to security information 4-44
maps
    changing configuration parameters 4-21
    creating and using multiple maps 4-41
    partitioning 4-20
    samples 4-46—4-47
    setting a home submap 4-38
    setting map groups and permissions 4-46
    submap modifications 4-37
    using read-only maps 4-43
modifying and viewing entity attributes 4-15—4-19
NetRanger Director user interface menu functions 4-23
    determining the version of a remote daemon 4-26
    remotely configuring NetRanger daemons 4-23
    resolving an alarm's IP address 4-26
    shunning IP addresses and Class C IP networks 4-26
    unshunning IP addresses and Class C IP networks 4-27
    viewing alarm context information 4-23
    viewing database status and configuration 4-25
    viewing event lists 4-24
network security maps 1-15
NSX management 1-16
NSX monitoring 1-15
post-installation 3-19
    configuring NetRanger background processes 3-19
    configuring network management background processes 3-20
pre-installation
    HP-UX and Sun Solaris 3-12
    IBM AIX 3-15
saving object data to a file 4-27
setting user groups 4-45
setup options 2-13
starting the Director 4-3
starting the NetRanger background processes 4-3
starting the network management background processes 4-3

starting the network management user interface 4-4
stopping the NetRanger background processes 4-5
stopping the network management user interface 4-4
submap hierarchy 4-6—4-9
submap modifications 4-37
symbol modifications 4-36
symbols
hiding 4-39
repositioning 4-39
searching for symbols 4-37
troubleshooting
Director not running A-2
Director running A-4
uninstallation G-1

E

encrypted sleeve
capabilities 1-13
introduction 1-3
event signatures 4-60—4-82
IP options events 4-65—4-67
TCP/IP event signatures 4-63
content-based signatures 4-64
context-based signatures 4-63
eventd 4-55—4-60
advanced setup
changing the event error notification user 4-60
monitoring multiple organizations 4-59
applications E-3
basic setup 4-56
event script setup 4-56
eventd.conf setup 4-58
overview 4-55
smid.conf setup 4-58
and user-defined actions 1-20
eventd.conf setup 4-58

F

File 5-46
FileMgmt tokens 5-46
filter points 1-7
installation of filters 3-8

filter templates B-3
filters
  DPF B-10
  first.fil B-3
  incom(X).fil B-3
  installing under NetSentry 3-8
  last.fil B-6
  shun.fil B-6
filters.cmd B-2
first.fil B-3
  default setting 1-7

H

hierarchies and message propagation 1-12
hosts E-15
how to partition a map 4-20

I

ICMP events 4-67—4-78
incom(X).fil B-3
install command E-7
intrusion detection engine 1-2
intrusion detection signatures C-2

L

last.fil B-6
limiting access to security information 4-44
local installation of Oracle D-2
loggerd
  applications E-3
  NumberOfSwitchBytes 5-7
  NumberOfSwitchMinutes 5-7
  SwitchFile 5-9

M

managed
  applications E-4
  configuration H-1

- location H-1
- managed.conf E-9
- and network devices H-1—H-7
- shunning H-7—H-12
- tokens H-15
- writing files H-14

managed.conf E-9
maps
- changing configuration parameters 4-21
- creating and using multiple maps 4-41
- partitioning 4-20
- setting a home submap 4-38
- setting map groups and permissions 4-46
- submap modifications 4-37

message propagation 1-12
modifying a submap 4-37
monitoring multiple organizations 4-59

N

NetRanger
- architecture 4-2
- bridging B-7
- default alarm settings C-1—C-8
- Director (see Director)
- intrusion detection signatures C-2
- physical installation considerations 2-13
- power requirements 2-13
- pre-installation 2-1
- token order in nrConfigure 4-51
- verify NetRanger/Oracle connectivity D-7

NetRanger Configuration Instructions and Worksheets 3-26
NetRanger installation
- analyze current network architecture 2-1
- BorderGuard/Passport installation (see BorderGuard)
- configuration
  - ICMP 3-28
  - TCP 3-29
  - UDP 3-31
- define all entry and exit points 2-1
- defining security policy 3-27
- Director installation (see Director)
- and firewalls 2-3
- gathering network and security information 3-28
- NSX installation (see NSX)

and security filters in existing routers 2-3
NetSentry 1-7
filter installation 3-8
signatures 4-81
NetSentry-based signatures 4-81
network protocols 1-6
new signatures 4-61
non-Oracle install and setup D-7
nr_log_alarm 5-38
nr_log_alarm_1 5-38
nr_log_command 5-40
nr_log_context 5-39
nr_log_error 5-40
nr_log_tcpconn 5-39
nrConfigure 1-17, 4-48—4-54
architecture 4-48
configuring an application 4-50
help 4-50
overview 4-48
quitting 4-49
starting 4-49
token order 4-51
nrdirmap
disabling 4-44
finding out about nrdirmap 4-27
nrdirmap's command line parameters 4-30—4-35
-a, Alarm consolidation threshold 4-30
-c, Critical value threshold 4-30
-d, Disable nrdirmap by default for new maps 4-31
examples 4-35
-f, force Full synchronization 4-31
-i, seconds to be Idle before sleep 4-32
-k, Keep symbol copies during delete 4-32
-m, Marginal value threshold 4-32
-o, maxiumum number of Objects represented in a map 4-32
-p, Propagate to IP Map 4-33
-s, Secure map creation 4-33
-t, Tracing enabled 4-34
nrstart E-7
nrstatus E-7
nrstop E-7
NSX
attack responses 1-8
attack signatures 1-5
capabilities 1-5
configuration

sysconfig-nsx utility 3-4
configuration commands E-6—E-7
daemon applications E-2
configd E-3
eventd E-3
loggerd E-3
managed E-4
postofficed E-4
sapd E-5
sensord E-5
smid E-5
data collection and SAP 1-18
event notification 4-1
file structure E-1
~/bin E-2
~/classes E-8
~/etc E-8
~/tmp E-19
first.fil 1-7
hardware components F-1
information generation 1-11
installation 3-1
attach cables 3-2
default root password 3-4
initial access and system configuration
console access 3-3
modem access 3-3
network access 3-2
serial access 3-3
logging in 3-4
position the NSX 3-1
installation options
installation on corporate network 2-11
installation on isolated network 2-10
installation on switched Ethernet network 2-12
introduction 2-9
introduction 1-2
intrusion detection 1-7
logging 1-9
physical and operational specifications 2-15
and real-time intrusion detection 1-7
sensord alarm generation 1-9
system commands E-7
system files E-14
auths E-16
daemons E-17

destinations E-16
hosts E-15
organizations E-14
routes E-17
services E-15
troubleshooting A-7
NSX 1000 1-7
NSX 2000 1-7
NSX 5000 1-7
NSX monitoring 1-15
NumberOfSwitchBytes 5-7
NumberOfSwitchMinutes 5-7

O

Oracle
create user D-6
install and setup D-1
installation and mounting D-2
local D-2
remote D-2
non-Oracle install and setup D-7
planning your environment D-1
troubleshooting A-8
UNIX setup of Oracle environment variables D-3—D-4
verify generic connectivity D-4—D-5
verify NetRanger/Oracle connectivity D-7
organizations E-14

P

packet filter devices
capabilities 1-6
packet filtering device 1-2
Passport 1-6, 1-7
configuration 3-8
configuration of netrangr user account H-14
user account installation 3-9
physical installation considerations 2-13
policy violations C-6
post office subsystem 1-3
postofficed
applications E-4
and daemon communication 1-11

pre-installation 2-1
product capabilities 1-3
pubkeys.dpf B-10

R

regular expression-based signatures 4-78
remote configuration of daemons 4-23
remote installation of Oracle D-2
resolving an alarm's IP address 4-26
routes E-17

S

sapd 5-9
  applications E-5
sapx 5-11
  alternate loading procedures 5-42
  and database load setup 5-24
  and native Oracle schemas 5-37
  parameters 5-11
Security Analysis Package (SAP)
  data analysis 1-19
  fail-safe features 5-8—5-9
    file loss from overwriting 5-9
    runaway serialization 5-8
  function, configuration, and usage 5-4
  introduction 1-3
  log file serialization 1-9
  push-pull mechanism 1-18
Security Management Interface (SMI) 1-3
security policy, defining 3-27
sensord
  alarm generation 1-9
  applications E-5
  and IP Session logs 1-10
  shunning 1-9
sensord.conf E-10
serialization of log files 1-9
services E-15
shun.fil B-6
shunning 1-6
  manual reconfiguration of BorderGuard/Passport 1-9
  and sensord 1-9

shunning IP addresses and Class C IP networks 4-26
unshunning IP addresses and Class C IP networks 4-27
signatures
atomic 1-8
attack 1-5
composite 1-8
content 1-6
context 1-6
intrusion detection C-2
NetSentry-based 4-81
new 4-61
regular expression-based 4-78
TCP/IP content-based signatures 4-64
TCP/IP context-based signatures 4-63
skel 5-41
sleeves.cmd B-12
sleeves.dpf B-11
smid
applications E-5
smid.conf E-13
smid.conf setup 4-58
SQL
database information 5-37
default schemas 5-37
starting the Director 4-3
starting the NetRanger background processes 4-3
starting the network management background processes 4-3
startup file B-2
stopping the Director 4-4
stopping the NetRanger background processes 4-5
stopping the network management user interface 4-4
submap modifications 4-37
SwitchFile 5-9
sysconfig-nsx utility 3-4
system commands
install E-7
nrstart E-7
nrstatus E-7
nrstop E-7

T

TCP/IP event signatures 4-63
tokens
FileMgmt 5-46

managed H-15
troubleshooting
connectivity A-6
Director not running A-2
Director running A-4
NSX A-7
Oracle A-8

U

uninstalling the Director G-1

V

verify NetRanger/Oracle connectivity D-7
viewing alarm context information 4-23

WheelGroup Corporation

WheelGroup Corporation welcomes your ideas on how to improve our documentation. Please take a moment to answer the questions below by checking either the "Yes" or "No" box. Explain any "No" responses in the sections provided. Include other comments in the "Comments" section. When you have finished, please detach and mail this page, or fax it to (210) 494-6303. You may also e-mail us about documentation issues at documentation@wheelgroup.com.

Does this manual contain all the information you expected?	◊ Yes	◊ No
Are all documented procedures in the manual correct?	◊ Yes	◊ No
Is this manual easy to read and understand?	◊ Yes	◊ No
Are the examples and diagrams helpful?	◊ Yes	◊ No
Are there enough examples and diagrams?	◊ Yes	◊ No
Other comments:		